!		n p ——
'	(P)	' nnnn —— p
(	(P)	(ssss)
*		n n —— p
*/		n n p —— q
+		n n —— q
+!		n p ——
+LOOP	(C)	n ——
,		n ——
-		n n —— q
—)	(P)	
.		n ——
/		n n —— q
/MOD		n n —— r q
0(		n —— f
0=		n —— f True if n is zero.
0)		n —— f True if n positive and non-zero.
1+		n —— q q=n+1
:		: nnnn
;	(C,P)	
;:	(C,P)	
;CODE	(C,P)	
;S	(E)	
(		n n —— f True if n(n
=		n n —— f True if n=n
)		n n —— f True if n)n
)R	(C)	n ——
?		p ——
@		p —— q
ABORT		
ABS		n —— q
AND		n n —— q
B!		n p ——
B@		p —— q
BASE	(U)	—— p
BEGIN	(CO+,P)	BEGIN ... WHILE ... REPEAT
BLK	(U)	—— p
BLOCK		b —— p
BMOVE		p q n ——
BYTE		p —— q
CELL		p —— q
CODE		CODE nnnn
CONSTANT		n CONSTANT nnnn ——
COUNT		p --- n n
CR		
DECIMAL		
DO	(C)	n n ——
DP+!		n ——
DPL	(U)	—— p
DROP		n ——
DUMP		n n ——
DUP		n —— n n
ELSE	(C2,P)	
END	(C2-,P)	f ——
ERASE-CORE		
FIND		FIND nnnn —— p
FLD		—— p
FLUSH		
FORGET		FORGET nnnn
HERE	(U)	—— p
HEX		

```
I       (C)          -- n
IF      (C2+,P)  f IF (true part) ELSE (false part) THEN --
IMMEDIATE
LAST                 -- p
LINE                 n -- p
LIST                 b --
LIT     (C)          LIT n
LOAD                 b --
LOOP    (C)
MAX                  n n -- q
MIN                  n n -- q
MINUS                n -- -n
MOD                  n n -- r
MOVE                 p q n --
NUMBER
O.                   n --
OCTAL
OR                   n n -- q
OVER                 n n -- n n n
REPEAT  (C2-,P)
ROT                  n n p -- n p n
SP@                  -- p
SPACE
SPACES               n --
SWAP                 n n -- n n
THEN    (C0-,P)
TYPE                 n n --
UPDATE
VARIABLE             n VARIABLE nnnn --
WHILE   (C2+,P)      f WHILE --
WORD                 c WORD pppp --
```

FORTH-77 EXTENSIONS

CUR	-- p		T	
MAP0	-- p		T	
READ-MAP			T	
REWIND			T	
+BLOCK	n -- b		X	
." (P)	." ssss"		X	39
1+!	p --		X	
1-	n -- q	q=n-1	X	15
1-!	p --		X	
2*	n -- q	q=n*2	X	15
2/	n -- q	q=n/2	X	15
2DROP	n n --		X	39
2DUP	n n -- n n n n		X	
2SWAP	n n p q -- p q n n		X	
(=	n n -- f True if n(n or n=n		X	
(>	n n -- f True if n unequal to n		X	
)=	n n -- f True if n)n or n=n		X	
ARRAY	n ARRAY nnnn --		X	
ASSEMBLER (P)			X	
BELL			X	
BUFFER	b -- p		X	
CASE (C2+,P)	n n -- (n)		X	
CHAIN	CHAIN vvvv		X	32
COM	n -- q		X	15
CONTEXT (U)	-- p		X	
CONTINUED (E)	b --		X	
CURRENT (U)	-- p		X	
DEFINITIONS	(vvvv) DEFINITIONS		X	
DISCARD			X	
DLIST (E)	n --		X	42
EDIT	b --		X	
EDITOR (P)			X	
EDOC			X	
FORTH (P)			X	
IFEND (E)			X	
IFTRUE (E)	f IFTRUE...OTHERWISE...IFEND --		X	
J (C)	-- n		X	31
K (C)	-- n		X	31
LEAVE (C)			X	
LINELOAD	n b --		X	
MS	n --		X	
NAND	n n -- q		X	
NOR	n n -- q		X	
NOT	n -- f		X	
OTHERWISE (E)			X	
PAGE			X	
PICK	n -- q		X	15
R) (C)	-- n		X	
REMEMBER	REMEMBER nnnn		X	
ROLL	u(n) u(n-1)...u(1) n -- u(n-1)...u(1) u(n)		X	
SET	n p SET nnnn --		X	16
U(	n n -- f		X	
U(=			X	
U)	Unsigned comparisons, in which the entire (16-bit)		X	
U)=	values are regarded as positive integers and compared.		X	
VOCABULARY (E)	VOCABULARY vvvv		X	
WRITE	n n --		X	37
WHERE			X	
XOR	n n -- q		X	15
[(P)			X	
]			X	
^	n p -- q		X	

This description of the FORTH standard and extended vocabularies was edited by the Standards Team of the FORTH USERS GROUP.
(Subject to amendments and approval planned for September 1978.)

Each glossary entry is annotated in its top right corner, to indicate its level of standardization (no mark implying it is standard) as follows:

T	Tape systems only.
X	Part of an extension to FORTH-77

The words are presented in the order of their numerically sorted identifier codes. The first line of each entry shows a symbolic description of the action of the word: Symbols indicating which parameters are to be placed on the stack before executing the word, 3 dashes (---) indicating execution, then any parameters left on the stack by the word. In this notation, the top of the stack is to the right. If the place of the word in the input string is not completely obvious, it is shown explicitly. If no dashes are shown the word does not affect the stack. Symbols are used as follows:

b	Block number
c	7-bit ASCII character code.
f	Flag : 0=False, non-zero=True. All words which return a flag return 0=False or 1=True.
m n p q r	16-bit integers.
nnnn pppp	The name of a word.
ssss	A string of characters.

Immediately following the name of a word, certain characters may appear within parentheses. These denote some special action or characteristics:

C	The word may be used only within a colon-definition A following digit (C0 or C2) indicates the number of memory cells used when the word is compiled if other than one. A following + or - sign indicates that the word either pushes a value onto the stack or removes one from the stack during compilation. (This action is not related to its action during execution and may be implementation dependent.)
E	The word may not normally be compiled within a colon-definition.
P	The word has its precedence bit set ; it is executed directly, even when encountered during compile mode.
U	The word applies to a user variable (one for which in multi-user systems each user has his own copy.)

Unless stated otherwise, all references to numbers apply to 16-bit integers, with the most significant bit as the sign bit and the negative in two's complement form. Similarly, all arithmetic will be assumed to be 16-bit signed integer arithmetic with error conditions and overflow indication unspecified.

The 1977 Standard definitions and Extensions.

! m p ---
Store m at address p.

' (P) ' nnnn --- p
Leave the address of the parameter field of nnnn
piler directive, ' is executed when encountered i
definition: the address of the following word's p
field is found immediately (at compilation) and s
the dictionary (after the address of LIT) as a li
be placed on the stack at execution time. Within
definition, ' nnnn is identical to: LIT ['

((P) (ssss)
Ignore a comment that will be delimited by a righ
thesis.

***** m n --- p
16-bit signed integer multiply. p=m*n

***/** m n p --- q
Leave q=m*n/p . Retention of an intermediate 31-b
duct permits greater accuracy than the otherwise
lent sequence: m n * p /

+ m n --- q
16-bit integer addition. q=m+n

+! m p ---
Add integer m to value at address p.

+BLOCK m --- b
Leave the sum of m plus the number of the block c
being interpreted.

+LOOP (C) m ---
Add m to the loop index. Exit from the loop is m
the resultant index reaches or passes the limit,
greater than zero; or when the index is less than
the limit, if m is less than zero. m may be a v

, m ---
Store m into the next available dictionary cell,
the dictionary pointer.

- m n --- q
16-bit integer subtraction: q=m-n

--) (P)
(Pronounced "nextblock") Continue interpretation
next block (Equivalent to 1 +BLOCK CONTINUED).

. m ---
Print the value on the stack as an integer, comm
according to the current number base.

." (P) ." ssss"
Transmit a message delimited by " to the select
device. The maximum number of characters may be
[illegible].

d e s c r i p t i o n o f t h e F O R T H
e r d a n d e x t e n d e d v o c a b u l a r i e s
d i t e d b y t h e S t a n d a r d s T e a m
e F O R T H U S E R S G R O U P .
o amendments and approval planned for September 1978.)

ry entry is annotated in its top right corner, to
its level of standardization (no mark implying it is
as follows:

Tape systems only.
Part of an extension to FORTH-77

are presented in the order of their numerically sorted
codes. The first line of each entry shows a symbolic
on of the action of the word: Symbols indicating which
s are to be placed on the stack before executing the
shes (--) indicating execution, then any parameters
e stack by the word. In this notation, the top of the
to the right. If the place of the word in the input
not completely obvious, it is shown explicitly. If no
e shown the word does not affect the stack. Symbols are
ollows:

Block number
7-bit ASCII character code.
Flag : 0=False, non-zero=True. All words which return a flag return 0=False or 1=True.
q r 16-bit integers.
pppp The name of a word.
A string of characters.

ly following the name of a word, certain characters may
ithin parentheses. These denote some special action or
istics:

The word may be used only within a colon-definition
A following digit (C0 or C2) indicates the number of memory cells used when the word is compiled if other than one. A following + or - sign indicates that the word either pushes a value onto the stack or removes one from the stack during compilation. (This action is not related to its action during execution and may be implementation dependent.)
The word may not normally be compiled within a colon-definition.
The word has its precedence bit set ; it is executed directly, even when encountered during compile mode.
The word applies to a user variable (one for which in multi-user systems each user has his own copy.)

tated otherwise, all references to numbers apply to 16-
ers, with the most significant bit as the sign bit and
tive in two's complement form. Similarly, all arithmetic
ssumed to be 16-bit signed integer arithmetic with er-
itions and overflow indication unspecified.

The 1977 Standard definitions and Extensions.

! m p --
Store m at address p.

' (P) ' nnnn -- p
Leave the address of the parameter field of nnnn . A compiler directive, ' is executed when encountered in a colon definition: the address of the following word's parameter field is found immediately (at compilation) and stored in the dictionary (after the address of LIT) as a literal to be placed on the stack at execution time. Within a colon definition, ' nnnn is identical to: LIT [' nnnn ,]

((P) (ssss)
Ignore a comment that will be delimited by a right parenthesis.

***** m n -- p
16-bit signed integer multiply. p=m*n

***/** m n p -- q
Leave q=m*n/p . Retention of an intermediate 31-bit product permits greater accuracy than the otherwise equivalent sequence: m n * p /

+ m n -- q
16-bit integer addition. q=m+n

+! m p --
Add integer m to value at address p.

+BLOCK m -- b X
Leave the sum of m plus the number of the block currently being interpreted.

+LOOP (C) m --
Add m to the loop index. Exit from the loop is made when the resultant index reaches or passes the limit, if m is greater than zero; or when the index is less than (passes) the limit, if m is less than zero. m may be a variable.

, m --
Store m into the next available dictionary cell, advancing the dictionary pointer.

- m n -- q
16-bit integer subtraction: q=m-n

--> (P)
(Pronounced "nextblock") Continue interpretation with the next block (Equivalent to 1 +BLOCK CONTINUED).

. m --
Print the value on the stack as an integer, converted according to the current number base.

." (P) ." ssss" X
Transmit a message delimited by " to the selected output device. The maximum number of characters may be an installation dependent value.

/ m n -- q
16-bit signed integer divide, m/n. The quotient is truncated ; any remainder is lost.

/MOD m n -- r q
16-bit integer divide, m/n. The quotient is left on top of the stack, the remainder beneath. The remainder has the sign of the dividend, m.

0< m -- f
Leave a true flag if m is negative

0= m -- f True if m is zero.

0> m -- f True if m positive and non-zero.

1+ m -- q q=m+1

1+! p -- X
Add 1 to the contents of location p

1- m -- q q=m-1 X

1-! p -- X
Subtract 1 from the contents of location p.

2* m -- q q=m*2 X

2/ m -- q q=m/2 X

2DROP m n -- X
Drop the top two values from the stack (to drop a double-precision number for example).

2DUP m n -- m n m n X
Duplicate the top two values on the stack.

2SWAP m n p q -- p q m n X
Swap two pairs of values (e.g. double-precision numbers).

: : nnnn
Create a dictionary entry defining nnnn as equivalent to the following sequence of Forth Words. Set compilation mode. (Extension: Set the context vocabulary equivalent to the current vocabulary.)

; (C,P)
Terminate a colon-definition and stop compilation.

;: (C,P)
Terminate a defining word nnnn, which can be subsequently executed to define a new word pppp. Subsequent use of pppp will cause the words between ;: and ; to be executed with the contents of the first parameter of pppp on the stack.

;CODE (C,P)
Stop compilation and terminate a new defining word nnnn. (Extension: Set the context vocabulary to Assembler.) When nnnn is executed to define a new word pppp , the execution address of pppp will point to the assembler sequence following the ;CODE of nnnn. Subsequent use of pppp will cause this machine-code sequence to be executed. A previously defined defining word must be included in the sequence of Forth Words preceding ;CODE.

;S (E)
Stop interpretation of a symbolic block.

< m n -- f True if m<n

<= m n -- f True if m<n or m=n

= m n -- f True if m=n

<> m n -- f True if m unequal to n

> m n -- f True if m>n

>= m n -- f True if m>n or m=n

>R (C) m --
Push m onto the top of the return stack. See R>

? p --
Print the value contained at address p in free fo
according to the current base.

@ p -- q
Leave the contents q of memory location p.

ABORT
Enter the abort sequence, clear the stacks and re
trol to the terminal, possibly printing an abort

ABS m -- q
Leave the obsolute value of a number.

AND m n -- q
Bitwise logical AND of m and n.

ARRAY m ARRAY nnnn --
Define an array named nnnn and allocate m+1 unini
locations onto the dictionary. The sequence i nn
leave the address of the i-th cell on the stack.
dex should be in the range 0 <= i <= m, but no ch
made for values exceeding this range.

ASSEMBLER (P)
Switch the context pointer so that dictionary sea
will begin at the Assembler Vocabulary. The Assem
cabulary is always chained to the current vocabul

B! m p --
Store the least significant 8 bits of m at byte-a

B@ p -- q
Return the 8-bit byte q found at byte-address p.

BASE (U) -- p
A variable containing the current number conversi

BEGIN (CO+,P) BEGIN ... WHILE ... REPEAT
or BEGIN ... END
Mark the start of a sequence of words to be execu
titively. If ... WHILE ... REPEAT is used the lo
be repeated as long as the top stack value encoun
WHILE is TRUE (REPEAT merely effects an unconditi
back to BEGIN); when WHILE sees a FALSE value (0)

m n -- q

16-bit signed integer divide, m/n. The quotient is truncated ; any remainder is lost.

m n -- r q

16-bit integer divide, m/n. The quotient is left on top of the stack, the remainder beneath. The remainder has the sign of the dividend, m.

m -- f

Leave a true flag if m is negative

m -- f True if m is zero.

m -- f True if m positive and non-zero.

m -- q q=m+1

p -- X

Add 1 to the contents of location p

m -- q q=m-1 X

p -- X

Subtract 1 from the contents of location p.

m -- q q=m*2 X

m -- q q=m/2 X

m n -- X

Drop the top two values from the stack (to drop a double-precision number for example).

m n -- m n m n X

Duplicate the top two values on the stack.

m n p q -- p q m n X

Swap two pairs of values (e.g. double-precision numbers).

: nnnn

Create a dictionary entry defining nnnn as equivalent to the following sequence of Forth Words. Set compilation mode. (Extension: Set the context vocabulary equivalent to the current vocabulary.)

; (C,P)

Terminate a colon-definition and stop compilation.

;: (C,P)

Terminate a defining word nnnn, which can be subsequently executed to define a new word pppp. Subsequent use of pppp will cause the words between ;: and ; to be executed with the contents of the first parameter of pppp on the stack.

;CODE (C,P)

Stop compilation and terminate a new defining word nnnn. (Extension: Set the context vocabulary to Assembler.) When nnnn is executed to define a new word pppp , the execution address of pppp will point to the assembler sequence following the ;CODE of nnnn. Subsequent use of pppp will cause this machine-code sequence to be executed. A previously defined defining word must be included in the sequence of Forth Words preceding ;CODE.

;S (E)

Stop interpretation of a symbolic block.

< m n -- f True if m<n

<= m n -- f True if m<n or m=n X

= m n -- f True if m=n

<> m n -- f True if m unequal to n X

> m n -- f True if m>n

>= m n -- f True if m>n or m=n X

>R (C) m --

Push m onto the top of the return stack. See R> .

? p --

Print the value contained at address p in free format, according to the current base.

@ p -- q

Leave the contents q of memory location p.

ABORT

Enter the abort sequence, clear the stacks and return control to the terminal, possibly printing an abort message.

ABS m -- q

Leave the absolute value of a number.

AND m n -- q

Bitwise logical AND of m and n.

ARRAY m ARRAY nnnn -- X

Define an array named nnnn and allocate m+1 uninitialed locations onto the dictionary. The sequence i nnnn will leave the address of the i-th cell on the stack. The index should be in the range 0 <= i <= m, but no check is made for values exceeding this range.

ASSEMBLER (P) X

Switch the context pointer so that dictionary searches will begin at the Assembler Vocabulary. The Assembler Vocabulary is always chained to the current vocabulary.

B! m p --

Store the least significant 8 bits of m at byte-address p

B@ p -- q

Return the 8-bit byte q found at byte-address p.

BASE (U) -- p

A variable containing the current number conversion base.

BEGIN (CO+,P) BEGIN ... WHILE ... REPEAT
or BEGIN ... END

Mark the start of a sequence of words to be executed repetitively. If ... WHILE ... REPEAT is used the loop will be repeated as long as the top stack value encountered by WHILE is TRUE (REPEAT merely effects an unconditional jump back to BEGIN); when WHILE sees a FALSE value (0) on the

stack it causes an immediate exit out of the loop. In case the sequence can be written such that the test for completion is at the end ... END can be used conveniently to end the loop on a TRUE value or to go back to BEGIN on FALSE. Both WHILE and END drop the value they test.

BELL X

Activate terminal bell or noisemaker as appropriate to device.

BLK (U) -- p

A variable containing the number of the block being listed or edited.

BLOCK b -- p

Leave the first address of Block b. If the block is not already in memory, it is transferred from disk or tape into whichever core buffer has been least recently accessed. If the block occupying that buffer has been updated, it is rewritten on disk or tape before Block b is read into the buffer.

BMOVE p q n --

Move the n bytes starting at byte-address p into the n byte-cells starting at byte-address q. The contents of p is moved first (OCTAL 40 2001 ! 2001 2002 100 BMOVE would put blanks into byte locations 2001 through 2100).

BUFFER b -- p X

Obtain a core buffer for Block b, leaving the first buffer cell address. The block is not read from disk, and is automatically marked as updated.

BYTE p -- q

Return the byte address of the first byte in memory cell p

CASE (C24,P) m n -- (m) X

m n CASE (action for m=n) ELSE (drop) THEN

If m equals n, drop both m and n and execute the words directly following CASE until the next ELSE or THEN ; otherwise, drop n but leave m and execute the words after ELSE (or THEN if no ELSE is used). The selection of one of many cases can be done by:

```
m n1 CASE (action for m=n1) ELSE
  n2 CASE (action for m=n2) ELSE
  n3 CASE (action for m=n3)
     ELSE (otherwise action) THEN THEN THEN
```

CELL p -- q

Return the word address of the memory cell holding byte p.

CHAIN CHAIN vvvv X

Connect the current vocabulary to all definitions that might be entered into vocabulary vvvv in the future. The current vocabulary may not be FORTH or ASSEMBLER. Any given vocabulary may only be chained once, but may be the object of any number of chainings. For example, every user-defined vocabulary may include the sequence, CHAIN FORTH.

CODE CODE nnnn

Create a dictionary entry defining nnnn as equivalent to the following sequence of assembler code. (Extension: set the context vocabulary to Assembler.)

COM m -- q

Complement each bit of m (Leave one's complement

CONSTANT m CONSTANT nnnn --

Create a word which when executed pushes m onto
Since the "constant" m may be modified by the se
q ' nnnn ! it is oftentimes advantageous to def
iable as a constant, particularly if it is acces
often than it is modified.

CONTEXT (U) -- p

A variable containing a pointer to the vocabular
dictonary searches are to begin. See CURRENT .

CONTINUED (E) b --

Continue interpretation at Block b. (The preferr
mentation in multi-buffer systems is such that t
buffer currently being accessed will be used for
of block b, leaving other buffers unaffected.)

COUNT p -- m n

Leave byte-address m and byte-count n of a messa
string beginning at word address p. It is presum
the first byte at p contains the byte-count and
actual message starts with the second byte in lo
Typically, COUNT is followed by WRITE or TYPE.

CR

Transmit carriage return/line feed codes to the
output device.

CUR -- p

A variable containing the physical record number
which the tape is currently positioned. REWIND s

CURRENT (U) -- p

A variable containing a pointer to the vocabular
which new words are to be entered. CURRENT @ @ l
link address of the next entry to be defined.

DECIMAL

Set the numeric conversion base to decimal mode.

DEFINITIONS (vvvv) DEFINITIONS

Set the current vocabulary (into which new defin
are placed) to vocabulary vvvv (the context voca
vvvv need not be specified explicitly.

DISCARD

A null-definition intended for use as a standard
word, as some version of DISCARD can always be f
the dictionary.

DLIST (E) m --

List the dictionary, starting at the entry begin
location m.

DO (C) m n --

Begin a loop, to be terminated by LOOP or +LOOP.
index begins at n, and may be modified at the en
loop by any positive or negative value. The loop
minated when an incremented index reaches or exc
or when a decremented index becomes less than m.

ack it causes an immediate exit out of the loop. In case e sequence can be written such that the test for comple- on is at the end ... END can be used conveniently to end e loop on a TRUE value or to go back to BEGIN on FALSE. th WHILE and END drop the value they test.

X

tivate terminal bell or noisemaker as appropriate to de- ice.

U) — p
variable containing the number of the block being listed edited.

b — p
ave the first address of Block b. If the block is not ready in memory, it is transferred from disk or tape to whichever core buffer has been least recently acces- d. If the block occupying that buffer has been updated, is rewritten on disk or tape before Block b is read to the buffer.

p q n —
ve the n bytes starting at byte-address p into the n te-cells starting at byte-address q. The contents of p moved first (OCTAL 40 2001 ! 2001 2002 100 BMOVE would t blanks into byte locations 2001 through 2100).

b — p X
tain a core buffer for Block b, leaving the first ffer cell address. The block is not read from disk, and automatically marked as updated.

p — q
turn the byte address of the first byte in memory cell p

C2+,P) m n — (m) X
m n CASE (action for m=n) ELSE (drop) THEN
f m equals n, drop both m and n and execute the words di- ctly following CASE until the next ELSE or THEN ; other- ise, drop n but leave m and execute the words after ELSE or THEN if no ELSE is used). The selection of one of many ses can be done by:

```
m n1 CASE (action for m=n1) ELSE
  n2 CASE (action for m=n2) ELSE
  n3 CASE (action for m=n3)
     ELSE (otherwise action) THEN THEN THEN
```

p — q
turn the word address of the memory cell holding byte p.

CHAIN vvvv X
nnect the current vocabulary to all definitions that ight be entered into vocabulary vvvv in the future. The rrent vocabulary may not be FORTH or ASSEMBLER. Any iven vocabulary may only be chained once, but may be the ject of any number of chainings. For example, every er-defined vocabulary may include the sequence, CHAIN RTH.

CODE nnnn
eate a dictionary entry defining nnnn as equivalent to he following sequence of assembler code. (Extension: set he context vocabulary to Assembler.)

COM m — q X
Complement each bit of m (Leave one's complement).

CONSTANT m CONSTANT nnnn —
Create a word which when executed pushes m onto the stack Since the "constant" m may be modified by the sequence: q ' nnnn ! it is oftentimes advantageous to define a variable as a constant, particularly if it is accessed more often than it is modified.

CONTEXT (U) — p X
A variable containing a pointer to the vocabulary in which dictionary searches are to begin. See CURRENT .

CONTINUED (E) b — X
Continue interpretation at Block b. (The preferred implementation in multi-buffer systems is such that the block buffer currently being accessed will be used for storage of block b, leaving other buffers unaffected.)

COUNT p — m n
Leave byte-address m and byte-count n of a message string string beginning at word address p. It is presumed that the first byte at p contains the byte-count and that the actual message starts with the second byte in location p. Typically, COUNT is followed by WRITE or TYPE.

CR
Transmit carriage return/line feed codes to the selected output device.

CUR — p T
A variable containing the physical record number before which the tape is currently positioned. REWIND sets CUR=1.

CURRENT (U) — p X
A variable containing a pointer to the vocabulary into which new words are to be entered. CURRENT @ @ leaves the link address of the next entry to be defined.

DECIMAL
Set the numeric conversion base to decimal mode.

DEFINITIONS (vvvv) DEFINITIONS X
Set the current vocabulary (into which new definitions are placed) to vocabulary vvvv (the context vocabulary). vvvv need not be specified explicitly.

DISCARD X
A null-definition intended for use as a standard REMEMBER word, as some version of DISCARD can always be found in the dictionary.

DLIST (E) m — X
List the dictionary, starting at the entry beginning at location m.

DO (C) m n —
Begin a loop, to be terminated by LOOP or +LOOP. The loop index begins at m, and may be modified at the end of the loop by any positive or negative value. The loop is terminated when an incremented index reaches or exceeds m, or when a decremented index becomes less than m. Within

a loop, the word I will place the current index value on the stack.
Within nested loops, the word I always returns the index of the innermost loop that is being executed, while J returns the index of the next outer loop, and K returns the index of the second outer loop.

DP+! n ——
Add the signed value n to the dictionary pointer (DP). As DP may be an internal register rather than a VARIABLE, it is accessible only through HERE and DP+! .

DPL (U) —— p
A variable containing a value indicating the number of digits following the fractional point in number output. A negative value at DPL indicates that no dot is to appear.

DROP m ——
Drop the top value from the stack.

DUMP m n ——
Dump the contents of n memory cells starting at address m. Normally, both addresses and contents are shown in the current base.

DUP m —— m m
Duplicate the top value on the stack

EDIT b ——
The Editor vocabulary is loaded, if not already in the dictionary, becoming the context vocabulary. Block b is listed. X

EDITOR (P)
The name of the Editor Vocabulary. If that vocabulary is loaded, EDITOR establishes it as the context vocabulary, thereby making its definitions accessible. X

ENDC
End of code ; terminate a code definition. Restore the context vocabulary pointer back to the current vocabulary. X

ELSE (C2,P)
Precede the false part of an IF..ELSE..THEN conditional. It may be omitted if the false part is empty.

END (C2-,P) f ——
Mark the end of a BEGIN-END loop. If f is true the loop is terminated. If f is false, control returns to the first word after the corresponding BEGIN.

ERASE-CORE
Mark all block-buffers as empty. Updated blocks are not flushed. Contents of buffers are subsequently undefined.

FIND FIND nnnn —— p
Return the compilation address of nnnn (i.e. the address that would normally be compiled when nnnn is encountered in a colon-definition) if that name can be found in the dictionary ; zero otherwise.

FLD —— p
A variable containing the field length reserved for a number during output conversion.

FLUSH
Write all blocks that have been flagged as "updat
disk or tape. Return when output is completed.

FORGET FORGET nnnn
Delete nnnn and all dictionary entries following
(Extension: Though nnnn must be in the context vo
to be found, the words that follow it are deleted
ter what vocabulary they belong to.

FORTH (P)
The name of the primary vocabulary. Execution mak
the context vocabulary. Since FORTH cannot be cha
anything, it becomes the only vocabulary that is
for dictionary entries.
Unless additional user vocabularies are defined,
definitions normally become part of the Forth Voc

HERE (U) —— p
Return the address of the next available dictiona
tion.

HEX
Switch the numeric conversion base to hexadecimal

I (C) —— m
Return the index of an innermost DO-loop.

IF (C2+,P) f IF (true part) ELSE (false part) THEN
f IF (true part) THEN ——
IF is the first word of a conditional. If f is tr
words following IF are executed and the words fol
ELSE are not executed. The ELSE part of the condi
is optional. If f is false, words between IF and
between IF and THEN when no ELSE is used, are ski
ELSE-THEN conditionals may be nested.

IFEND (E)
Terminate a conditional interpretation sequence b
IFTRUE.

IFTRUE (E) f IFTRUE...OTHERWISE...IFEND ——
Unlike IF-ELSE-THEN, these conditionals may be em
during interpretation. In conjunction with the wo
] they may be used within a colon-definition to
compilation, although they are not to be compiled
words cannot be nested.

IMMEDIATE
Mark the most recently made dictionary entry such
when encountered at compile time it will be execu
rather than compiled.

J (C) —— m
Within a nested DO-loop, return the index of the
outer loop.

K (C) —— m
Within a nested DO-loop, return the index of the
outer loop.

LAST —— p
A variable containing the compilation address of

loop, the word I will place the current index value on
e stack.
ithin nested loops, the word I always returns the index
f the innermost loop that is being executed, while J
eturns the index of the next outer loop, and K returns
e index of the second outer loop.

n ——

dd the signed value n to the dictionary pointer (DP). As
P may be an internal register rather than a VARIABLE, it
s accessible only through HERE and DP+! .

(U) —— p

variable containing a value indicating the number of di-
its following the fractional point in number output. A
egative value at DPL indicates that no dot is to appear.

m ——

rop the top value from the stack.

m n ——

ump the contents of n memory cells starting at address m.
ormally, both addresses and contents are shown in the
urrent base.

m —— m m

uplicate the top value on the stack

b —— X

he Editor vocabulary is loaded, if not already in the
ictionary, becoming the context vocabulary. Block b is
isted.

(P) X

he name of the Editor Vocabulary. If that vocabulary is
oaded, EDITOR establishes it as the context vocabulary,
hereby making its definitions accessible.

X

nd of code ; terminate a code definition. Restore the
ontext vocabulary pointer back to the current vocabulary.

(C2,P)

recede the false part of an IF..ELSE..THEN conditional.
t may be omitted if the false part is empty.

(C2-,P) f ——

ark the end of a BEGIN-END loop. If f is true the loop
s terminated. If f is false, control returns to the first
ord after the corresponding BEGIN.

ORE

ark all block-buffers as empty. Updated blocks are not
lushed. Contents of buffers are subsequently undefined.

FIND nnnn —— p

eturn the compilation address of nnnn (i.e. the address
hat would normally be compiled when nnnn is encountered
n a colon-definition) if that name can be found in the
ictionary ; zero otherwise.

—— p

variable containing the field length reserved for a num-
er during output conversion.

FLUSH

Write all blocks that have been flagged as "updated" to disk or tape. Return when output is completed.

FORGET FORGET nnnn

Delete nnnn and all dictionary entries following it. (Extension: Though nnnn must be in the context vocabulary to be found, the words that follow it are deleted no matter what vocabulary they belong to.

FORTH (P) X

The name of the primary vocabulary. Execution makes FORTH the context vocabulary. Since FORTH cannot be chained to anything, it becomes the only vocabulary that is searched for dictionary entries.
Unless additional user vocabularies are defined, new user definitions normally become part of the Forth Vocabulary.

HERE (U) —— p

Return the address of the next available dictionary location.

HEX

Switch the numeric conversion base to hexadecimal.

I (C) —— m

Return the index of an innermost DO-loop.

IF (C2+,P) f IF (true part) ELSE (false part) THEN ——
f IF (true part) THEN ——

IF is the first word of a conditional. If f is true, the words following IF are executed and the words following ELSE are not executed. The ELSE part of the conditional is optional. If f is false, words between IF and ELSE, or between IF and THEN when no ELSE is used, are skipped. IF-ELSE-THEN conditionals may be nested.

IFEND (E) X

Terminate a conditional interpretation sequence begun by IFTRUE.

IFTRUE (E) f IFTRUE...OTHERWISE...IFEND —— X

Unlike IF-ELSE-THEN, these conditionals may be employed during interpretation. In conjunction with the words [and] they may be used within a colon-definition to control compilation, although they are not to be compiled. These words cannot be nested.

IMMEDIATE

Mark the most recently made dictionary entry such that when encountered at compile time it will be executed rather than compiled.

J (C) —— m X

Within a nested DO-loop, return the index of the next outer loop.

K (C) —— m X

Within a nested DO-loop, return the index of the second outer loop.

LAST —— p

A variable containing the compilation address of the most

recently made dictionary entry, which may not yet be a complete or valid entry .

LEAVE (C) X
Force termination of a DO-loop at the next opportunity by setting the loop limit equal to the current value of the index. The index itself remains unchanged, and execution proceeds normally until LOOP or +LOOP is encountered.

LINE m -- p
Leave the word address of the beginning of line m for the block whose number is contained at BLK. (For editing purposes a block is divided in 16 lines, numbered 0-15, of 64 characters.)

LINELOAD m b -- X
Begin interpreting at line m of Block b. (0 =< m =< 15)

LIST b --
List the ASCII symbolic contents of Block b on the selected output device.

LIT (C) LIT m
Automatically compiled before each literal encountered in a colon-definition. Execution of LIT causes the contents of the next dictionary cell to be pushed onto the stack. NOTE: On the CDC 6400, LIT is implemented so as to access the next 15-bit parcel of a 60-bit word. This implementation is recommended for other machines of large word size

LOAD b --
Begin interpretation of Block b. The block must terminate its own interpretation with ;S , --) (nextblock) or CONTINUED.

LOOP (C)
Increment the DO-loop index by one, terminating the loop if the new index is equal to or greater than the limit.

MAP0 -- p T
Return the address of the first location in the tape map.

MAX m n -- q
Leave the greater of two numbers.

MIN m n -- q
Leave the lesser of two numbers.

MINUS m -- -m
Negate a number by taking its two's complement.

MOD m n -- r
Leave the remainder of m/n, with the same sign as m.

MOVE p q n --
Move the contents of n memory cells beginning at address p into n cells beginning at address q. The contents of p is moved first ; overlapping of data can occur.
(0 10 ! 10 11 4 MOVE clears locations 10 through 15).

MS n -- X
Delay for approximately n milliseconds.

NAND m n -- q
Logical AND followed by COMplement.

NOR m n -- q
Logical OR followed by COMplement.

NOT m -- f
Equivalent to 0=

NUMBER
Convert a character string left in the dictionar
by WORD as a number, returning the result in reg
internal temporary locations, or on the stack. T
rance of characters that cannot be properly inte
will cause an error exit appropriate to the inst

0. m --
Convert and output in octal mode, unsigned, and
by a blank. BASE is unchanged. Format specificat
observed.

OCTAL
Set the number-conversion base to octal.

OR m n -- q
Bitwise logical inclusive OR of m and n.

OTHERWISE (E)
An interpreter-level conditional word . See IFTR

OVER m n -- m n m
Push the second stack value.

PAGE
Clears the terminal screen or performs an action
to the output device currently active.

PICK n -- q
Return the nth value on the stack, not counting
(2 PICK is equivalent to OVER)

R) (C) -- n
Pop the top value from the return stack and push
the user stack. See)R.

READ-MAP
Read to the next file mark on tape, constructing
in memory (the map) relating physical block posi
logical block numbers. The tape should normally
to its load point before executing READ-MAP.

REMEMBER REMEMBER nnnn
Define a word nnnn which, when executed, will ca
and all subsequently defined words to be deleted
dictionary. The word nnnn may be compiled into a
ted from a colon-definition. The sequence DISCAR
DISCARD provides a standardized preface to any g
of transient blocks.

REPEAT (C2-,P)
Effect an unconditional jump back to the beginni
BEGIN ... WHILE ... REPEAT loop. See BEGIN.

ecently made dictionary entry, which may not yet be a complete or valid entry .

(C) X
orce termination of a DO-loop at the next opportunity by etting the loop limit equal to the current value of the ndex. The index itself remains unchanged, and execution roceeds normally until LOOP or +LOOP is encountered.

m --- p
eave the word address of the beginning of line m for the lock whose number is contained at BLK. (For editing purposes a block is divided in 16 lines, numbered 0-15, of 64 characters.)

D m b --- X
egin interpreting at line m of Block b. (0 =(m =(15)

b ---
ist the ASCII symbolic contents of Block b on the selected output device.

(C) LIT m
utomatically compiled before each literal encountered in colon-definition. Execution of LIT causes the contents of the next dictionary cell to be pushed onto the stack. OTE: On the CDC 6400, LIT is implemented so as to access he next 15-bit parcel of a 60-bit word. This implementation is recommended for other machines of large word size

b ---
egin interpretation of Block b. The block must terminate its own interpretation with ;S , --) (nextblock) or CONTINUED.

(C)
ncrement the DO-loop index by one, terminating the loop if the new index is equal to or greater than the limit.

--- p T
eturn the address of the first location in the tape map.

m n --- q
eave the greater of two numbers.

m n --- q
eave the lesser of two numbers.

m --- -m
egate a number by taking its two's complement.

m n --- r
eave the remainder of m/n, with the same sign as m.

p q n ---
ove the contents of n memory cells beginning at address into n cells beginning at address q. The contents of p is moved first ; overlapping of data can occur.
(0 10 ! 10 11 4 MOVE clears locations 10 through 15).

n --- X
elay for approximately n milliseconds.

NAND m n --- q X
Logical AND followed by COMplement.

NOR m n --- q X
Logical OR followed by COMplement.

NOT m --- f X
Equivalent to 0=

NUMBER
Convert a character string left in the dictionary buffer by WORD as a number, returning the result in regis 6nn internal temporary locations, or on the stack. The appearance of characters that cannot be properly interpreted will cause an error exit appropriate to the installation.

0. m ---
Convert and output in octal mode, unsigned, and preceded by a blank. BASE is unchanged. Format specifications are observed.

OCTAL
Set the number-conversion base to octal.

OR m n --- q
Bitwise logical inclusive OR of m and n.

OTHERWISE (E) X
An interpreter-level conditional word . See IFTRUE.

OVER m n --- m n m
Push the second stack value.

PAGE X
Clears the terminal screen or performs an action suitable to the output device currently active.

PICK n --- q X
Return the nth value on the stack, not counting n itself. (2 PICK is equivalent to OVER)

R) (C) --- n X
Pop the top value from the return stack and push it onto the user stack. See >R.

READ-MAP T
Read to the next file mark on tape, constructing a table in memory (the map) relating physical block positions to logical block numbers. The tape should normally be rewound to its load point before executing READ-MAP.

REMEMBER REMEMBER nnnn X
Define a word nnnn which, when executed, will cause nnnn and all subsequently defined words to be deleted from the dictionary. The word nnnn may be compiled into and executed from a colon-definition. The sequence DISCARD REMEMBER DISCARD provides a standardized preface to any group of transient blocks.

REPEAT (C2-,P)
Effect an unconditional jump back to the beginning of a BEGIN ... WHILE ... REPEAT loop. See BEGIN.

REWIND T

Rewind the tape to its load point, setting CNR=1.

ROLL u(n) u(n-1)...u(1) n --- u(n-1)...u(1) u(n) X

Extract the nth value from the stack, leaving it on top and moving the remaining values into the vacated position (3 ROLL is equivalent to ROT ; 1 ROLL is a null operation 0 ROLL is undefined.)

ROT m n p --- n p m

Rotate the top three values on the stack, bringing the deepest to the top.

SET m p SET nnnn --- X

Define a word nnnn which, when executed, will cause the value m to be stored at address p.

SP@ --- p

Return the address of the top of the stack.
(E.g. 1 2 SP@ @ . . . would type 2 2 1).

SPACE

Output a space character to the selected output device.

SPACES n ---

Output n spaces to the selected output device. No action taken for n < 1.

SWAP m n --- n m

Exchange the top two stack values.

THEN (CO-,P)

Terminate an ..IF...ELSE...THEN conditional sequence.

TYPE m n ---

Send a string of n characters starting at byte address m, to the terminal.

U< m n --- f X
U<= X
U) Unsigned comparisons, in which the entire (16-bit) X
U>= values are regarded as positive integers and compared. X

UPDATE

Flag the most-recently referenced block as updated. The block will subsequently be transferred automatically to disk or tape should its buffer be required for storage of a different block. See FLUSH.

VARIABLE m VARIABLE nnnn ---

Create a word nnnn which, when executed, will push the address of a variable (initialized to m) onto the stack.

VOCABULARY (E) VOCABULARY vvvv X

Define a vocabulary. Subsequent use of vvvv will make vvvv the context vocabulary. The sequence vvvv DEFINITIONS will make vvvv the current vocabulary, into which definitions are placed.

WHILE (C2+,P) f WHILE ---

Test the value on the stack and, if FALSE, exit out of a BEGIN ... WHILE ... REPEAT loop. See BEGIN.

WRITE m n ---

Output the n characters starting at byte address selected output device.

WHERE

Output information about the status of Forth (E.g an error abort). Indicate at least the last word and the last block accessed.

WORD c WORD pppp ---

Read the next word from the input string being in until a delimiter c is found, storing the packed string beginning at the next available dictionary (HERE) with the character count in the first byte

XOR m n --- q

Bitwise logical exclusive OR of m and n.

[(P)

Stop compilation. The words following the left br a colon-definition are executed, not compiled.

]

Resume compilation. Following words are compiled dictionary

^ n p --- q

Compute q as n to the power p. All are signed int

----------end of THE FORTH-77 GLOSSARY----------

T

ewind the tape to its load point, setting CUR=1.

u(n) u(n-1)...u(1) n --- u(n-1)...u(1) u(n) X

xtract the nth value from the stack, leaving it on top nd moving the remaining values into the vacated position (3 ROLL is equivalent to ROT ; 1 ROLL is a null operation 0 ROLL is undefined.)

m n p --- n p m

otate the top three values on the stack, bringing the eepest to the top.

m p SET nnnn --- X

efine a word nnnn which, when executed, will cause the alue m to be stored at address p.

--- p

eturn the address of the top of the stack. (E.g. 1 2 SP@ @ . . . would type 2 2 1).

utput a space character to the selected output device.

n ---

utput n spaces to the selected output device. No action aken for n < 1.

m n --- n m

xchange the top two stack values.

(CO-,P)

erminate an ..IF...ELSE...THEN conditional sequence.

m n ---

end a string of n characters starting at byte address , to the terminal.

m n --- f X
X

nsigned comparisons, in which the entire (16-bit) X
alues are regarded as positive integers and compared. X

lag the most-recently referenced block as updated. The lock will subsequently be transferred automatically to isk or tape should its buffer be required for storage of different block. See FLUSH.

E m VARIABLE nnnn ---

reate a word nnnn which, when executed, will push the ad- ress of a variable (initialized to m) onto the stack.

ARY (E) VOCABULARY vvvv X

efine a vocabulary. Subsequent use of vvvv will make vvvv he context vocabulary. The sequence vvvv DEFINITIONS will ake vvvv the current vocabulary, into which definitions re placed.

(C2+,P) f WHILE ---

est the value on the stack and, if FALSE, exit out of a EGIN ... WHILE ... REPEAT loop. See BEGIN.

WRITE m n --- X

Output the n characters starting at byte address m to the selected output device.

WHERE X

Output information about the status of Forth (E.g. after an error abort). Indicate at least the last word compiled and the last block accessed.

WORD c WORD pppp ---

Read the next word from the input string being interpreted until a delimiter c is found, storing the packed character string beginning at the next available dictionary location (HERE) with the character count in the first byte.

XOR m n --- q X

Bitwise logical exclusive OR of m and n.

[(P) X

Stop compilation. The words following the left bracket in a colon-definition are executed, not compiled.

] X

Resume compilation. Following words are compiled into the dictionary

^ n p --- q X

Compute q as n to the power p. All are signed integers.

----------end of THE FORTH-77 GLOSSARY----------

1 CODE DEFINITIONS

This vocabulary allows the assembly of HP 2100 series machine code instructions into FORTH words called "code definitions". Code definitions start with the word CODE which constructs a dictionary entry of which the code address points to the parameter field, where the machine instructions are assembled, and end with a return to the inner interpreter (NEXT). When a word thusly constructed is to be executed the inner interpreter jumps directly to the machine code sequence defined. This code then has complete control of the machine until it does the jump to NEXT.

There is an important difference between the "compilation" of a normal FORTH word with a colon definition and the "assembly" of a code definition. During compilation FORTH is in the compile state so that words with zero precedence are not interpreted but compiled. During assembly FORTH remains in the execute state and it is the execution of the words from the assembler vocabulary that effects the assembly (and the loading) into the dictionary.

2 STACK HANDLING, ACCESS TO POINTERS AND REGISTERS.

The following words are used to access the basic registers:

A — pushes onto the stack the value 0. This may then be used to refer to memory location zero, i.e. the A-register.

B — pushes the address of location 1, i.e. the B-register.

SP — pushes the address of the memory locaton which holds the address of the current top of stack.

SP1 — pushes the address of the memory location which holds the address of the word just below the top of stack.

) — converts an address already on the stack to an indirect address (by setting bit 15).

A) — A)

B) — B)

S) — SP)

S1) — SP1)

DP — pushes the address of the location which holds the address of the next available dictionary location

HERE — pushes the value of the dictionary pointer. Equiv. to DP @

IC — pushes the address of the inner interpreter pointer.

, — pops a word from the stack and pushes it onto the dictionary (incrementing the dictionary pointer).

‡ — replaces a value on the stack by the address of a location holding that value. If necessary, a new location is added to the base page link and constant table.

3 CODE TERMINATORS

A code definition may be (acceptably) terminated by one following 7 words:

NEXT, pushes onto the dictionary (assembles) a jump to tine which selects the next FORTH word to be exec

The remaining 6 terminators all assemble exactly one ju to a different routine. All of these routines proceed t same routine (NEXT). Their run-time actions are given b

PUT, replace top of stack with the contents of the A-r

PUSH, push contents of A-register onto the stack.

POP, pop one word from the stack.

BINARY, replace top two words on stack by contents of

2POP, pop two words from the stack.

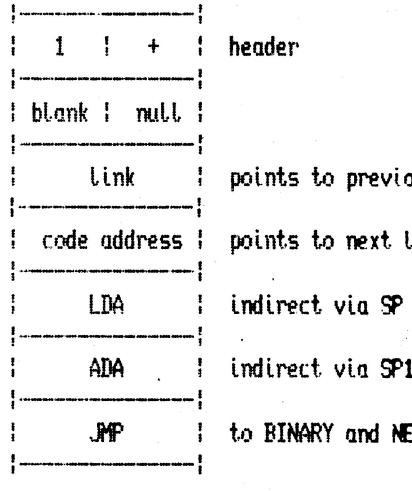
2BINARY, replace top three words on stack by contents o

4 EXAMPLE OF A CODE DEFINITION

The line of code

```
CODE +   S) LDA,  S1) ADA,  BINARY
```

puts the following into the dictionary:



The individual actions of the 6 words used to define "

CODE constructs header, including link and code addres

S) pushes an indirect reference (via SP) to the top

LDA, assembles a "load A" instruction for which the ad taken from the stack.

S1) pushes an indirect reference to the word below th

ADA, assembles "add to A" , popping the address from t

BINARY, assembles a jump to the routine that will repl words on the stack by the contents of A and and return to the inner interpreter (NEXT).

DEFINITIONS

cabulary allows the assembly of HP 2100 series machine
structions into FORTH words called "code definitions".
finitions start with the word CODE which constructs a
ary entry of which the code address points to the parame-
ld, where the machine instructions are assembled, and end
return to the inner interpreter (NEXT). When a word thus-
tructed is to be executed the inner interpreter jumps di-
to the machine code sequence defined. This code then has
e control of the machine until it does the jump to NEXT.

s an important difference between the "compilation" of a
FORTH word with a colon definition and the "assembly" of
definition. During compilation FORTH is in the compile
o that words with zero precedence are not interpreted but
d. During assembly FORTH remains in the execute state and
he execution of the words from the assembler vocabulary
ffects the assembly (and the loading) into the dictionary.

K HANDLING, ACCESS TO POINTERS AND REGISTERS.

llowing words are used to access the basic registers:

ushes onto the stack the value 0. This may then be used
o refer to memory location zero, i.e. the A-register.

ushes the address of location 1, i.e. the B-register.

ushes the address of the memory locaton which holds the
ddress of the current top of stack.

ushes the address of the memory location which holds the
ddress of the word just below the top of stack.

converts an address already on the stack to an indirect
ddress (by setting bit 15).

A)

B)

SP)

SP1)

ushes the address of the location which holds the address
of the next available dictionary location

ushes the value of the dictionary pointer. Equiv. to DP @

ushes the address of the inner interpreter pointer.

ops a word from the stack and pushes it onto the dictio-
ary (incrementing the dictionary pointer).

eplaces a value on the stack by the address of a location
olding that value. If necessary, a new location is added
o the base page link and constant table.

3 CODE TERMINATORS

A code definition may be (acceptably) terminated by one of the following 7 words:

NEXT, pushes onto the dictionary (assembles) a jump to the routine which selects the next FORTH word to be executed.

The remaining 6 terminators all assemble exactly one jump, each to a different routine. All of these routines proceed to the same routine (NEXT). Their run-time actions are given below.

PUT, replace top of stack with the contents of the A-register.

PUSH, push contents of A-register onto the stack.

POP, pop one word from the stack.

BINARY, replace top two words on stack by contents of A .

2POP, pop two words from the stack.

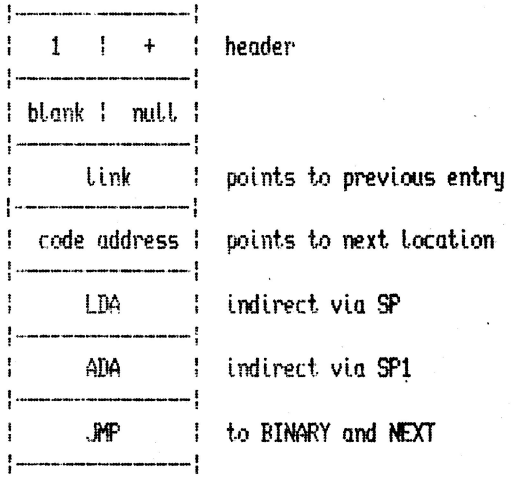
2BINARY, replace top three words on stack by contents of A .

4 EXAMPLE OF A CODE DEFINITION

The line of code

```
CODE +   S) LDA,  S1) ADA,  BINARY,
```

puts the following into the dictionary:



The individual actions of the 6 words used to define " + " are:

CODE constructs header, including link and code address.

S) pushes an indirect reference (via SP) to the top of stack.

LDA, assembles a "load A" instruction for which the address is taken from the stack.

S1) pushes an indirect reference to the word below the top.

ADA, assembles "add to A" , popping the address from the stack.

BINARY, assembles a jump to the routine that will replace two words on the stack by the contents of A and and that will return to the inner interpreter (NEXT).

5 INSTRUCTION CODES

The assembler uses mnemonics for a subset of the base instruction set described in the chapter on programming information in the Hewlett Packard 2100 or 21MX computer reference manual (with which the user is presumed to be familiar). Generally, the FORTH mnemonic is that of the manufacturer with a comma appended, to emphasize the action of storing into the dictionary.

Composite mnemonics for the most popular combinations of the so-called micro-instructions have been replaced with single names (e.g. ALS, for ALF,ALF) and a simple mechanism exists to define new mnemonics. In fact, the full power of FORTH is available to create special codes, pseudo-operations and macro-instructions.

As in the HP manual the base set of instructions is divided by format into 5 categories. For each group there is a special defining word (i.e. a word whose execution results in the definition of another) to define the members of that group.

5.1 - Memory Reference Instructions

MR m MR nnnn ---
Defines a word which can assemble a mem. ref. instruction.

Execution of a word defined by MR requires an address, q, on the stack, and results in the assembly of an instruction with m in its instruction code field and q in its address field. The indirect bit concurs with bit 15 of q. A base page link may be generated if q is not in page 0 or the current page.

All words in this group are named as HP's mnemonic with a comma appended:

ADA,	add to A
ADB,	add to B
AND,	"and" to A
CPA,	compare to A
CPB,	compare to B
IOR,	"inclusive or" to A
ISZ,	increment and skip if zero
JMP,	jump
JSB,	jump to subroutine
LDA,	load A
LDB,	load B
STA,	store A
STB,	store B
XOR,	"exclusive or" to A

5.2 - Register Reference Instructions

CPU n CPU nnnn ---
Defines nnnn to be a word which can assemble a register reference instruction whose value is simply n .

The implemented subset is shown below in FORTH source format with the Hewlett-Packard equivalent for the deviating mnemonics

```
1000 CPU ALS,   ( A left shift          )
5000 CPU BLS,   ( B left shift          )
2100 CPU CLE,   ( clear E               )
1600 CPU ELA,   ( rotate E left with A  )
  22 CPU RAL,   ( rotate A left         )
4022 CPU RBL,   ( rotate B left         )
2400 CPU CLA,   ( clear A               )
6400 CPU CLB,   ( clear B               )
3000 CPU CMA,   ( complement A          )
7000 CPU CMB,   ( complement B          )
2004 CPU INA,   ( increment A           )
6004 CPU INB,   ( increment B           )
2001 CPU RSS,   ( unconditional skip      RSS )
2002 CPU A0,    ( skip if A is zero       SSA )
6002 CPU B0,    ( skip if B is zero       SSB )
2040 CPU E0,    ( skip if E is zero       SEZ )
2020 CPU A+,    ( skip if A is positive   SSA )
6020 CPU B+,    ( skip if B is positive   SSB )
2021 CPU A-,    ( skip if A is negative   SSA,RSS
6021 CPU B-,    ( skip if B is negative   SSB,RSS
1727 CPU AL8,   ( shift A left 8 bits     ALF,ALF
5727 CPU BL8,   ( shift B left 8 bits     BLF,BLF
1222 CPU 2RAL,  ( rotate A left 2 bits    RAL,RAL
5222 CPU 2RBL,  ( rotate B left 2 bits    RBL,RBL
1121 CPU 2ARS,  ( shift A right 2 bits    ARS,ARS
5121 CPU 2BRS,  ( shift B right 2 bits    BRS,BRS
  65 CPU CAER,  ( rotate A right, bit 0
                (   lsb to E, 0 to msb    CLE,ERA
3004 CPU TCA,   ( two's complement of A   CMA,INA
7004 CPU TCB,   ( two's complement of B   CMB,INB
```

The word NOT , if used after one of the conditional ski
modifies the instruction to reverse the skip condition.

5.3 - Input/Output instructions

I/O n I/O nnnn ---
Defines a word which can assemble an I/O instruct

When nnnn is executed it takes the device select code,
the stack and assembles n and p into the appropriate fi

CLC,	clear control
CLF,	clear flag
LIA,	load into A
LIB,	load into B
OTA,	output A
OTB,	output B
SFC,	skip if flag clear
SFS,	skip if flag set
STC,	set control
STF,	set flag

The word CLF may be used after CLC, LIA, LIB, OTA, OTB,
to modify the instruction such thot it will also clear

5.4 - Extended Arithmetic Memory Reference Instructions

DEAU n DEAU nnnn ---
Defines a word which, given an address q on the s
assemble a two word instruction with q in the sec

MPY,	multiply	DLD,	double loa
DIV,	divide	DST,	double sto

When these instructions are not available (e.g. on the
they may be macro's referring to apprpriate subroutines

RUCTION CODES

embler uses mnemonics for a subset of the base instruc-
t described in the chapter on programming information in
Lett Packard 2100 or 21MX computer reference manual (with
he user is presumed to be familiar). Generally, the FORTH
c is that of the manufacturer with a comma appended, to
ze the action of storing into the dictionary.

te mnemonics for the most popular combinations of the so-
micro-instructions have been replaced with single names
L8, for ALF,ALF) and a simple mechanism exists to define
onics. In fact, the full power of FORTH is available to
special codes, pseudo-operations and macro-instructions.

he HP manual the base set of instructions is divided by
into 5 categories. For each group there is a special defi
rd (i.e. a word whose execution results in the definition
her) to define the members of that group.

emory Reference Instructions

m MR nnnn ---

efines a word which can assemble a mem. ref. instruction.

on of a word defined by MR requires an address, q, on the
and results in the assembly of an instruction with m in
truction code field and q in its address field. The indi-
t concurs with bit 15 of q. A base page link may be gene-
f q is not in page 0 or the current page.

ds in this group are named as HP's mnemonic with a comma
d:

DA, add to A
DB, add to B
ND, "and" to A
PA, compare to A
PB, compare to B
OR, "inclusive or" to A
SZ, increment and skip if zero
MP, jump
SB, jump to subroutine
DA, load A
DB, load B
TA, store A
TB, store B
OR, "exclusive or" to A

egister Reference Instructions

n CPU nnnn ---

efines nnnn to be a word which can assemble a register
eference instruction whose value is simply n .

lemented subset is shown below in FORTH source format
e Hewlett-Packard equivalent for the deviating mnemonics

```
1000 CPU ALS,   ( A left shift          )
5000 CPU BLS,   ( B left shift          )
2100 CPU CLE,   ( clear E               )
1600 CPU ELA,   ( rotate E left with A  )
  22 CPU RAL,   ( rotate A left         )
4022 CPU RBL,   ( rotate B left         )
2400 CPU CLA,   ( clear A               )
6400 CPU CLB,   ( clear B               )
3000 CPU CMA,   ( complement A          )
7000 CPU CMB,   ( complement B          )
2004 CPU INA,   ( increment A           )
6004 CPU INB,   ( increment B           )
2001 CPU RSS,   ( unconditional skip      RSS )
2002 CPU A0,    ( skip if A is zero       SSA )
6002 CPU B0,    ( skip if B is zero       SSB )
2040 CPU E0,    ( skip if E is zero       SEZ )
2020 CPU A+,    ( skip if A is positive   SSA )
6020 CPU B+,    ( skip if B is positive   SSB )
2021 CPU A-,    ( skip if A is negative   SSA,RSS )
6021 CPU B-,    ( skip if B is negative   SSB,RSS )
1727 CPU AL8,   ( shift A left 8 bits     ALF,ALF )
5727 CPU BL8,   ( shift B left 8 bits     BLF,BLF )
1222 CPU 2RAL,  ( rotate A left 2 bits    RAL,RAL )
5222 CPU 2RBL,  ( rotate B left 2 bits    RBL,RBL )
1121 CPU 2ARS,  ( shift A right 2 bits    ARS,ARS )
5121 CPU 2BRS,  ( shift B right 2 bits    BRS,BRS )
  65 CPU CAER,  ( rotate A right, bit 0           )
                (   lsb to E, 0 to msb    CLE,ERA )
3004 CPU TCA,   ( two's complement of A   CMA,INA )
7004 CPU TCB,   ( two's complement of B   CMB,INB )
```

The word NOT , if used after one of the conditional skip words, modifies the instruction to reverse the skip condition.

5.3 - Input/Output instructions

I/O n I/O nnnn ---

Defines a word which can assemble an I/O instruction.

When nnnn is executed it takes the device select code, p , from the stack and assembles n and p into the appropriate fields.

```
CLC,   clear control
CLF,   clear flag
LIA,   load into A
LIB,   load into B
OTA,   output A
OTB,   output B
SFC,   skip if flag clear
SFS,   skip if flag set
STC,   set control
STF,   set flag
```

The word CLF may be used after CLC, LIA, LIB, OTA, OTB, or STC, to modify the instruction such thot it will also clear the flag.

5.4 - Extended Arithmetic Memory Reference Instructions

DEAU n DEAU nnnn ---

Defines a word which, given an address q on the stack, can assemble a two word instruction with q in the second word.

```
MPY,  multiply          DLD,  double load
DIV,  divide            DST,  double store
```

When these instructions are not available (e.g. on the HP2114), they may be macro's referring to apprpriate subroutines

5.5 - Extended Arithmetic Register Reference Instructions

None implemented.

Note: New instructions need not be explicitly defined. For example, the 21MX "compare words" instruction could be written as:

105776 (CMW),

6 STRUCTURE IN CODE DEFINITIONS

Writing code should be a rare activity for the user, and is justified only for I/O handlers, time critical routines and new type definitions (with ;CODE). Routines over say 20 instructions long should really be an exception. The absence of labels therefore, represents less of a problem than the programmer used to conventional assemblers might initially think it would.

Labels perform two functions: to name stored constants or variables or to help control program flow. To perform the first function one can use named VARIABLEs, or one can write " HERE n , " before a code definition to store n in the dictionary and leave the address on the stack for later use with a memory reference code, or one can use a variant of CODE called ,CODE .

n ,CODE nnnn p1 , ... pn , --- constructs a dictionary entry for nnnn and, by setting the code address to the n+1th word of the parameter field, reserves n locations into which p1 ... pn are stored before the rest of the code is assembled.

Two "high-level" constructs are available for controlling flow. They are:

(condition test) IF, (true action) ELSE, (false action)THEN,

and

HERE (action1) (condition test) WHILE, (action 2) REPEAT,

These resemble the high level constructs .. IF ... ELSE ... THEN and BEGIN ... WHILE ... REPEAT both in name and in function. Their implementation is described below.

The condition test preceding IF, or WHILE, is a conditional skip If this is a register reference word it may be followed by NOT . The CPA, and CPB, conditional skips may not be followed by NOT . They are best read "A not equal" and "B not equal" respectively and for that reason are provided with the synonyms ANE, and BNE,

IF, leaves an empty place in the dictionary and pushes the address of that place onto the stack.

ELSE, assembles a jump to this point into the empty place left by IF, and replaces the top of stack with its own address to be processed by THEN, .

THEN, assembles a jump to this point into the empty place left by IF, or ELSE, popping the address off the stack.

HERE pushes the address of this point onto the stack.

WHILE, as IF,

REPEAT, assembles a jump to the next location i
empty space left by WHILE, and a jump t
address left by HERE into the present l
popping the two addresses off the stack.

Note that if in the repetitive construct action 2 is
can write

HERE (action) (condition test) JM

Considering that it is assembler code this is quite a
but keep in mind that a true condition (skip conditio
place) now means a loop exit rather than a repeat.

7 MACRO'S

An oddity of HP FORTH is that, because of the lack of
in the HP 2100 series, two stack pointers, SP and SP1,
maintained. The current implementation of the code te
still assumes that code definitions do not cause them
of step. Two words are defined to manipulate the stac

ISP, increments the two stack pointers (=
DSP, decrements the two stack pointers (=

ISP, assembles two instructions (ISZ and ISZ), whereas
assembles four and destroys the contents of the B regi

Other macro's are

SUB, subtract from A (3 instructions)
ADM, add A to memory (2 instructions)

8 SUBROUTINES

Code definitions cannot call on other FORTH words the
definitions can simply because during execution of the
definition FORTH is not in control.

However, the assembler does allow subroutines, specifi
lows

SUBROUTINE nnnn (assembler code) RETURN,

nnnn can now be called only from within code definitio
means of nnnn JSB, .

The word SUBROUTINE constructs an entry for nnnn as a
initializing its value to that of a HALT instruction.
checks the first parameter field of the last defined
entry. If that contains a HALT instruction it replaces
by a 0 and assembles an indirect jump to that locatio
wise it aborts with the message: RETURN, ? .

9 NEW TYPE DEFINITIONS

Please consult a FORTH manual for an explication of th
;CODE . For HP FORTH, one should know that NEXT leaves
register pointing to the code address field of the wor
executed. Incrementing B will point it to the first p

end of THE FORTH HP ASSEMBLER description

xtended Arithmetic Register Reference Instructions

plemented.

ew instructions need not be explicitly defined. For exam-
e 21MX "compare words" instruction could be written as:
105776 (CMW),

CTURE IN CODE DEFINITIONS

code should be a rare activity for the user, and is jus-
only for I/O handlers, time critical routines and new
finitions (with ;CODE). Routines over say 20 instructions
ould really be an exception. The absence of labels there-
epresents less of a problem than the programmer used to
ional assemblers might initially think it would.

perform two functions: to name stored constants or varia-
to help control program flow. To perform the first func-
e can use named VARIABLEs, or one can write " HERE n , "
a code definition to store n in the dictionary and leave
ress on the stack for later use with a memory reference
r one can use a variant of CODE called ,CODE .

nnnn p1 , ... pn , --- constructs a dictionary entry for
d, by setting the code address to the n+1th word of the
er field, reserves n locations into which p1 ... pn are
before the rest of the code is assembled.

gh-Level" constructs are available for controlling flow.
e:

ndition test) IF, (true action) ELSE, (false action)THEN,

E (action1) (condition test) WHILE, (action 2) REPEAT,

resemble the high level constructs .. IF ... ELSE ... THEN
GIN ... WHILE ... REPEAT both in name and in function.
mplementation is described below.

dition test preceding IF, or WHILE, is a conditional skip
is a register reference word it may be followed by NOT .
, and CPB, conditional skips may not be followed by NOT .
e best read "A not equal" and "B not equal" respectively
that reason are provided with the synonyms ANE, and BNE,

F, leaves an empty place in the dictionary and pushes the address of that place onto the stack.

LSE, assembles a jump to this point into the empty place left by IF, and replaces the top of stack with its own address to be processed by THEN, .

HEN, assembles a jump to this point into the empty place left by IF, or ELSE, popping the address off the stack.

ERE pushes the address of this point onto the stack.

HILE, as IF,

REPEAT, assembles a jump to the next location into the empty space left by WHILE, and a jump to the address left by HERE into the present location, popping the two addresses off the stack.

Note that if in the repetitive construct action 2 is empty, one can write

HERE (action) (condition test) JMP,

Considering that it is assembler code this is quite acceptable but keep in mind that a true condition (skip condition takes place) now means a loop exit rather than a repeat.

7 MACRO'S

An oddity of HP FORTH is that, because of the lack of indexing in the HP 2100 series, two stack pointers, SP and SP1, are maintained. The current implementation of the code terminators still assumes that code definitions do not cause them to be out of step. Two words are defined to manipulate the stack pointers

ISP, increments the two stack pointers (=POP)
DSP, decrements the two stack pointers (=PUSH)

ISP, assembles two instructions (ISZ and ISZ), whereas DSP, assembles four and destroys the contents of the B register.

Other macro's are

SUB, subtract from A (3 instructions)
ADM, add A to memory (2 instructions)

8 SUBROUTINES

Code definitions cannot call on other FORTH words the way colon definitions can simply because during execution of the code definition FORTH is not in control.

However, the assembler does allow subroutines, specified as follows

SUBROUTINE nnnn (assembler code) RETURN,

nnnn can now be called only from within code definitions by means of nnnn JSB, .

The word SUBROUTINE constructs an entry for nnnn as a variable, initializing its value to that of a HALT instruction. RETURN, checks the first parameter field of the last defined dictionary entry. If that contains a HALT instruction it replaces the HALT by a 0 and assembles an indirect jump to that location, otherwise it aborts with the message: RETURN, ? .

9 NEW TYPE DEFINITIONS

Please consult a FORTH manual for an explication of the word ;CODE . For HP FORTH, one should know that NEXT leaves the B-register pointing to the code address field of the word to be executed. Incrementing B will point it to the first parameter.

end of THE FORTH HP ASSEMBLER description

PRINTABLE ASCII CHARACTERS

OCT	DEC	HEX	CHAR	OCT	DEC	HEX	CHAR	OCT	DEC	HEX	CHAR
40	32	20		100	64	40	@	140	96	60	`
41	33	21	!	101	65	41	A	141	97	61	a
42	34	22	"	102	66	42	B	142	98	62	b
43	35	23	#	103	67	43	C	143	99	63	c
44	36	24	$	104	68	44	D	144	100	64	d
45	37	25	%	105	69	45	E	145	101	65	e
46	38	26	&	106	70	46	F	146	102	66	f
47	39	27	'	107	71	47	G	147	103	67	g
50	40	28	(	110	72	48	H	150	104	68	h
51	41	29	)	111	73	49	I	151	105	69	i
52	42	2A	*	112	74	4A	J	152	106	6A	j
53	43	2B	+	113	75	4B	K	153	107	6B	k
54	44	2C	,	114	76	4C	L	154	108	6C	l
55	45	2D	-	115	77	4D	M	155	109	6D	m
56	46	2E	.	116	78	4E	N	156	110	6E	n
57	47	2F	/	117	79	4F	O	157	111	6F	o
60	48	30	0	120	80	50	P	160	112	70	p
61	49	31	1	121	81	51	Q	161	113	71	q
62	50	32	2	122	82	52	R	162	114	72	r
63	51	33	3	123	83	53	S	163	115	73	s
64	52	34	4	124	84	54	T	164	116	74	t
65	53	35	5	125	85	55	U	165	117	75	u
66	54	36	6	126	86	56	V	166	118	76	v
67	55	37	7	127	87	57	W	167	119	77	w
70	56	38	8	130	88	58	X	170	120	78	x
71	57	39	9	131	89	59	Y	171	121	79	y
72	58	3A	:	132	90	5A	Z	172	122	7A	z
73	59	3B	;	133	91	5B	[	173	123	7B	{
74	60	3C	<	134	92	5C	\	174	124	7C	|
75	61	3D	=	135	93	5D	]	175	125	7D	}
76	62	3E	>	136	94	5E	^	176	126	7E	~
77	63	3F	?	137	95	5F	_	177	127	7F	▮